A/B Testing with Machine Learning - A Step-by-Step Tutorial

business-science.io/business/2019/03/11/ab-testing-machine-learning.html

11 March 2019

Written by Matt Dancho on March 11, 2019

With the rise of digital marketing led by tools including Google Analytics, Google Adwords, and Facebook Ads, a key competitive advantage for businesses is using A/B testing to determine effects of digital marketing efforts. Why? In short, small changes can have big effects.

This is why A/B testing is a huge benefit. A/B Testing enables us to determine whether changes in landing pages, popup forms, article titles, and other digital marketing decisions improve conversion rates and ultimately customer purchasing behavior. **A successful A/B Testing strategy can lead to massive gains - more satisfied users, more engagement, and more sales - Win-Win-Win**.

> A key competitive advantage for businesses is using A/B testing

A major issue with traditional, statistical-inference approaches to A/B Testing is that it only compares 2 variables - an experiment/control to an outcome. The problem is that **customer behavior is vastly more complex than this**. Customers take different paths, spend different amounts of time on the site, come from different backgrounds (age, gender, interests), and more. **This is where Machine Learning excels - generating insights from complex systems.**

Article Overview

In this article you will experience how to **implement Machine Learning for A/B Testing step-by-step**. After reading this post you will:

1. Understand what ***A/B Testing*** is

2. Understand why ***Machine Learning*** is a better approach for performing A/B Testing versus traditional statistical inference (e.g. z-score, t-test)

3. Get a ***Step-by-Step Walkthrough*** for implementing machine learning for A/B Testing in `R` using 3 different algorithms:
    - Linear Regression
    - Decision Trees
    - XGBoost
4. Develop a ***Story for what contributes to the goal*** of gaining Enrollments

5. Get a ***Learning Recommendation*** for those that want to learn how to implement machine learning following best practices for any business problem.

1.0 What is A/B Testing?

A/B Testing is a tried-and-true method commonly performed using a traditional ***statistical inference approach*** grounded in a hypothesis test (e.g. t-test, z-score, chi-squared test). In plain English, 2 tests are run in parallel:

1. **Treatment Group (Group A) -** This group is exposed to the new web page, popup form, etc.

2. **Control Group (Group B) -** This group experiences no change from the current setup.

The goal of the A/B is then to compare the conversion rates of the two groups using statistical inference.

The problem is that the world is not a vacuum involving only the experiment (treatment vs control group) and effect. The situation is vastly more complex and dynamic. Consider these situations:

- **Users have different characteristics**: Different ages, genders, new vs returning, etc

- **Users spend different amounts of time on the website**: Some hit the page right away, others spend more time on the site

- **Users are find your website differently**: Some come from email or newsletters, others from web searches, others from social media

- **Users take different paths**: Users take actions on the website going to different pages prior to being confronted with the event and goal

Often modeling an A/B test in this vacuum can lead to misunderstanding of the true story.

> The world is not a vacuum involving only the changes and effect. The situation is more complex.

This is where Machine Learning can help.

2.0 Why use Machine Learning?

Unlike statistical inference, Machine Learning algorithms enable us to model complex systems that include all of the ongoing events, user features, and more. There are a number of algorithms each with strengths and weaknesses.

> An attractive benefit to Machine Learning is that we can combine multiple approaches to gain insights.

Rather than discuss in abstract, we can use an example from Udacity's A/B Testing Course, but apply the applied Machine Learning techniques from our Business Analysis with R Course to gain better insights into the inner-workings of the system rather than simply comparing an experiment and control group in an A/B Test.

3.0 A/B Test Using Machine Learning: Step-By-Step Walkthrough

We'll ***implement machine learning*** to perform the A/B test using the `R` statistical programming language, an excellent tool for business professionals seeking to advance their careers by learning Data Science and Machine Learning [Read 6 Reasons to Learn R for Business Next].

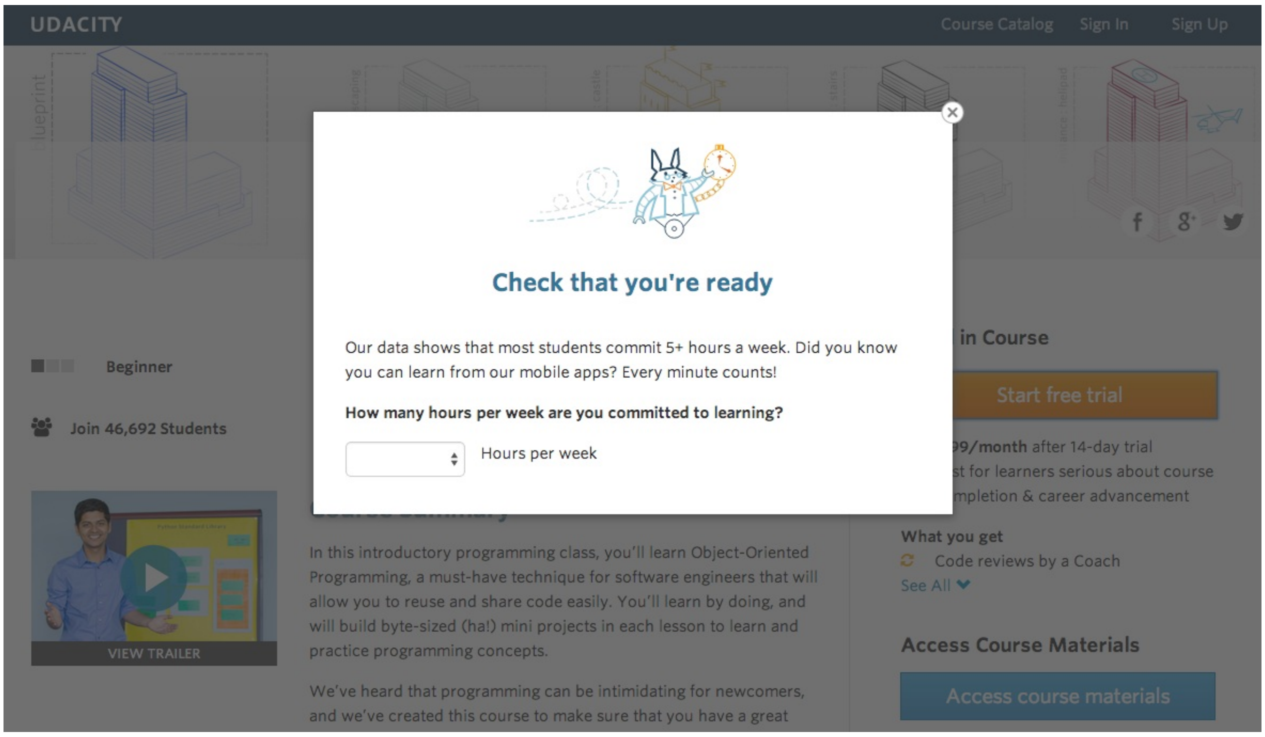
For those interested in the traditional statistical inference approach to A/B Testing, we found this article on Kaggle to be of very high quality: A/B Tests With Python

Experiment Name: "Free Trial" Screener

In the experiment, Udacity tested a change where if the student clicked "start free trial", they were asked how much time they had available to devote to the course.

If the student indicated 5 or more hours per week, they would be taken through the checkout process as usual. If they indicated fewer than 5 hours per week, a message would appear indicating that Udacity courses usually require a greater time commitment for successful completion.

This screenshot shows what the experiment looks like.


UDACITY

Course Catalog Sign In Sign Up

Check that you're ready

Our data shows that most students commit 5+ hours a week. Did you know you can learn from our mobile apps? Every minute counts!

How many hours per week are you committed to learning?

Hours per week

Beginner

Join 46,692 Students

VIEW TRAILER

In this introductory programming class, you'll learn Object-Oriented Programming, a must-have technique for software engineers that will allow you to reuse and share code easily. You'll learn by doing, and will build byte-sized (ha!) mini projects in each lesson to learn and practice programming concepts.

We've heard that programming can be intimidating for newcomers, and we've created this course to make sure that you have a great

in Course

Start free trial

99/month after 14-day trial

st for learners serious about course

mpletion & career advancement

What you get

Code reviews by a Coach

See All

Access Course Materials

Access course materials

Why Implement the Form?

The goal with this popup was that this might set clearer expectations for students upfront, thus reducing the number of frustrated students who left the free trial because they didn't have enough time.

However, what Udacity wants to avoid is "significantly" reducing the number of students that continue past the free trial and eventually complete the course.

Project Goal

In this analysis, we will investigate which features are contributing enrollments and determine if there is an impact on enrollments from the new "Setting Expectations" form.

- The users that experience the form will be denoted as "Experiment = 1"
- The control group (users that don't see the form) will be denoted as "Experiment = 0".

3.1 Get the Data

The data set for this A/B Test can be retrieved from Kaggle Data Sets.

- Control Data
- Experiment Data

3.2 Libraries

We'll need the following libraries for this tutorial. You can install them with `install.packages("package_name")`. We'll be using:

- `tidyverse` and `tidyquant`: These are the core data manipulation and visualization packages. We'll mainly be using `dplyr` for data manipulation, `ggplot2` for data visualization, and `tidyquant` themes for business reporting.
- `parsnip`, `rsample`, `recipes`, and `yardstick`: These are the tidyverse modeling packages. The `parsnip` package is an amazing tool that connects to the main machine learning algorithms. We teach `parsnip` in-depth (44 lessons, 5 hours of video) in Business Analysis with R, Week 6, Part 2 - Machine Learning (Regression).
- `rpart`, `rpart.plot`, and `xgboost`: These are the modeling libraries that we'll connect to through the `parsnip` interface.

```
# Core packages
library(tidyverse)
library(tidyquant)

# Modeling packages
library(parsnip)
library(recipes)
library(rsample)
library(yardstick)
library(broom)

# Connector packages
library(rpart)
library(rpart.plot)
library(xgboost)
```

3.3 Import the Data

Next, we can import the data using the `read_csv()` function. *Side Note* - We teach Data Import with `readr`, `odbc` (for data bases) in Week 2 of Business Analysis with R.

```
# Import data
control_tbl <- read_csv("data/control_data.csv")
experiment_tbl <- read_csv("data/experiment_data.csv")
```

3.4 Investigate the Data

We can use the `head()` function to see what the first 5 rows of data looks like. The pipe (`%>%`) is used to chain functions into complex expressions, a key feature of the `tidyverse`.

```
control_tbl %>% head(5)
```

```
## # A tibble: 5 x 5
##   Date        Pageviews Clicks Enrollments Payments
##   <chr>           <dbl>  <dbl>       <dbl>    <dbl>
## 1 Sat, Oct 11      7723    687         134       70
## 2 Sun, Oct 12      9102    779         147       70
## 3 Mon, Oct 13     10511    909         167       95
## 4 Tue, Oct 14      9871    836         156      105
## 5 Wed, Oct 15     10014    837         163       64
```

We have 5 columns consisting of:

- **Date**: a character formatted Day, Month, and Day of Month
- **Pageviews**: An aggregated count of Page Views on the given day
- **Clicks**: An aggregated count of Page Clicks on the given day for the page in question
- **Enrollments**: An aggregated count of Enrollments by day.
- **Payments**: An aggregated count of Payments by day.

Next, we inspect the "Control Group" using `glimpse()` the data to get an idea of the data format (column classes and data size).

```
control_tbl %>% glimpse()
```

```
## Observations: 37
## Variables: 5
## $ Date        <chr> "Sat, Oct 11", "Sun, Oct 12", "Mon, Oct 13", "T...
## $ Pageviews   <dbl> 7723, 9102, 10511, 9871, 10014, 9670, 9008, 743...
## $ Clicks      <dbl> 687, 779, 909, 836, 837, 823, 748, 632, 691, 86...
## $ Enrollments <dbl> 134, 147, 167, 156, 163, 138, 146, 110, 131, 16...
## $ Payments    <dbl> 70, 70, 95, 105, 64, 82, 76, 70, 60, 97, 105, 9...
```

Last, we can check the "Experiment" group (i.e. the Treatment Group) with `glimpse()` as well to make sure it's in the same format.

```
experiment_tbl %>% glimpse()
```

```
## Observations: 37
## Variables: 5
## $ Date        <chr> "Sat, Oct 11", "Sun, Oct 12", "Mon, Oct 13", "T...
## $ Pageviews   <dbl> 7716, 9288, 10480, 9867, 9793, 9500, 9088, 7664...
## $ Clicks      <dbl> 686, 785, 884, 827, 832, 788, 780, 652, 697, 86...
## $ Enrollments <dbl> 105, 116, 145, 138, 140, 129, 127, 94, 120, 153...
## $ Payments    <dbl> 34, 91, 79, 92, 94, 61, 44, 62, 77, 98, 71, 70,...
```

Key Points:

- 37 total observations in the control set and 37 in the experiment set

- Data is time-based and aggregated by day - This isn't the best way to understand complex user behavior, but we'll go with it

- We can see that `Date` is formatted as a character data type. This will be important when we get to data quality. We'll extract day of the week features out of it.

- Data between the experiment group and the control group is in the same format. Same number of observations (37 days) since the groups were tested in parallel.

3.5 Data Quality Check

Next, let's check the data quality. We'll go through a process involves:

- **Check for Missing Data** - Are values missing? What should we do?

- **Check Data Format** - Is data in correct format for analysis? Are all features created and in the right class?

We will use mainly `dplyr` in this section. *Side Note* - Data Manipulation with `dplyr` , `tidyr` , `lubridate` (time-series), `stringr` (text), and `forcats` (categorical) is taught in-depth (Weeks 2 and 3, 60+ lessons, 2 Challenges) in our Business Analysis with R course.

3.5.1 Check for Missing Data

The next series of operations calculates the count of missing values in each column with `map(~ sum(is.na(.)))` , converts to long format with `gather()` , and arranges descending with `arrange()` . *Side Note* - We teach these functions and techniques in Week 2 of Business Analysis with R course.

```
control_tbl %>%
   map_df(~ sum(is.na(.))) %>%
   gather(key = "feature", value = "missing_count") %>%
   arrange(desc(missing_count))
```

```
## # A tibble: 5 x 2
##   feature     missing_count
##   <chr>               <int>
## 1 Enrollments            14
## 2 Payments               14
## 3 Date                    0
## 4 Pageviews               0
## 5 Clicks                  0
```

Key Point: We have 14 days of missing observations that we need to investigate

Let's see if the missing data (`NA`) is consistent in the experiment set.

```
experiment_tbl %>%
   map_df(~ sum(is.na(.))) %>%
   gather(key = "feature", value = "missing_count") %>%
   arrange(desc(missing_count))
```

```
## # A tibble: 5 x 2
##   feature     missing_count
##   <chr>               <int>
## 1 Enrollments            14
## 2 Payments               14
## 3 Date                    0
## 4 Pageviews               0
## 5 Clicks                  0
```

Key Point: The count of missing data is consistent (a good thing). We still need to figure out what's going on though.

Let's see which values are missing using the `filter()`.

```
control_tbl %>%
    filter(is.na(Enrollments))
```

```
## # A tibble: 14 x 5
##    Date        Pageviews Clicks Enrollments Payments
##    <chr>           <dbl>  <dbl>       <dbl>    <dbl>
##  1 Mon, Nov 3       9437    788          NA       NA
##  2 Tue, Nov 4       9420    781          NA       NA
##  3 Wed, Nov 5       9570    805          NA       NA
##  4 Thu, Nov 6       9921    830          NA       NA
##  5 Fri, Nov 7       9424    781          NA       NA
##  6 Sat, Nov 8       9010    756          NA       NA
##  7 Sun, Nov 9       9656    825          NA       NA
##  8 Mon, Nov 10     10419    874          NA       NA
##  9 Tue, Nov 11      9880    830          NA       NA
## 10 Wed, Nov 12     10134    801          NA       NA
## 11 Thu, Nov 13      9717    814          NA       NA
## 12 Fri, Nov 14      9192    735          NA       NA
## 13 Sat, Nov 15      8630    743          NA       NA
## 14 Sun, Nov 16      8970    722          NA       NA
```

Key Point: We don't have Enrollment information from November 3rd on. We will need to remove these observations.

3.5.2 Check Data Format

We'll just check the data out, making sure its in the right format for modeling.

```
control_tbl %>% glimpse()
```

```
## Observations: 37
## Variables: 5
## $ Date        <chr> "Sat, Oct 11", "Sun, Oct 12", "Mon, Oct 13", "T...
## $ Pageviews   <dbl> 7723, 9102, 10511, 9871, 10014, 9670, 9008, 743...
## $ Clicks      <dbl> 687, 779, 909, 836, 837, 823, 748, 632, 691, 86...
## $ Enrollments <dbl> 134, 147, 167, 156, 163, 138, 146, 110, 131, 16...
## $ Payments    <dbl> 70, 70, 95, 105, 64, 82, 76, 70, 60, 97, 105, 9...
```

Key Points:

- Date is in character format. It doesn't contain year information. Since the experiment was only run for 37 days, we can only realistically use the "Day of Week" as a predictor.

- The other columns are all numeric, which is OK. We will predict the number of Enrollments (regression) (taught in Week 6 of Business Analysis with R course)

- Payments is an outcome of Enrollments so this should be removed.

3.6 Format Data

Now that we understand the data, let's put it into the format we can use for modeling. We'll do the following:

- Combine the `control_tbl` and `experiment_tbl`, adding an "id" column indicating if the data was part of the experiment or not
- Add a "row_id" column to help for tracking which rows are selected for training and testing in the modeling section
- Create a "Day of Week" feature from the "Date" column
- Drop the unnecessary "Date" column and the "Payments" column
- Handle the missing data (`NA`) by removing these rows.
- Shuffle the rows to mix the data up for learning
- Reorganize the columns

Here is the full transformation in one `dplyr` pipe. Notice that this entire series of operations is concise and readable. We teach how to do Data Manipulation and Cleaning throughout our Business Analysis with R course. This is the most important skill for a data scientist because it's where you will spend about 60%-80% of your time.

```
set.seed(123)
data_formatted_tbl <- control_tbl %>%

   # Combine with Experiment data
   bind_rows(experiment_tbl, .id = "Experiment") %>%
   mutate(Experiment = as.numeric(Experiment) - 1) %>%

   # Add row id
   mutate(row_id = row_number()) %>%

   # Create a Day of Week feature
   mutate(DOW = str_sub(Date, start = 1, end = 3) %>%
           factor(levels = c("Sun", "Mon", "Tue", "Wed",
                        "Thu", "Fri", "Sat"))
       ) %>%
   select(-Date, -Payments) %>%

   # Remove missing data
   filter(!is.na(Enrollments)) %>%

   # Shuffle the data (note that set.seed is used to make reproducible)
   sample_frac(size = 1) %>%

   # Reorganize columns
   select(row_id, Enrollments, Experiment, everything())

data_formatted_tbl %>% glimpse()

## Observations: 46
## Variables: 6
## $ row_id      <int> 14, 50, 18, 52, 54, 2, 22, 49, 21, 17, 53, 16, …
## $ Enrollments <dbl> 220, 122, 154, 127, 213, 147, 156, 128, 174, 23…
## $ Experiment  <dbl> 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,…
## $ Pageviews   <dbl> 9434, 8176, 9363, 8669, 9655, 9102, 8460, 9737,…
## $ Clicks      <dbl> 673, 642, 736, 669, 771, 779, 681, 801, 706, 75…
## $ DOW         <fct> Fri, Thu, Tue, Sat, Mon, Sun, Sat, Wed, Fri, Mo…
```

3.7 Training and Testing Sets

With the data formatted properly for analysis, we can now separate into training and testing sets using an 80% / 20% ratio. We can use the `initial_split()` function from `rsample` to create a split object, then extracting the `training()` and `testing()` sets.

```
set.seed(123)
split_obj <- data_formatted_tbl %>%
   initial_split(prop = 0.8, strata = "Experiment")

train_tbl <- training(split_obj)
test_tbl  <- testing(split_obj)
```

We can take a quick glimpse of the training data. It’s 38 observations randomly selected.

```
train_tbl %>% glimpse()
```

```
## Observations: 38
## Variables: 6
## $ row_id      <int> 14, 50, 18, 54, 2, 22, 49, 21, 17, 53, 38, 19, ...
## $ Enrollments <dbl> 220, 122, 154, 213, 147, 156, 128, 174, 233, 15...
## $ Experiment  <dbl> 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,...
## $ Pageviews   <dbl> 9434, 8176, 9363, 9655, 9102, 8460, 9737, 8890,...
## $ Clicks      <dbl> 673, 642, 736, 771, 779, 681, 801, 706, 759, 69...
## $ DOW         <fct> Fri, Thu, Tue, Mon, Sun, Sat, Wed, Fri, Mon, Su...
```

And, we can take a quick glimpse of the testing data. It's the remaining 8 observations.

```
test_tbl %>% glimpse()
```

```
## Observations: 8
## Variables: 6
## $ row_id      <int> 52, 16, 4, 59, 20, 44, 57, 13
## $ Enrollments <dbl> 127, 161, 156, 142, 167, 127, 207, 127
## $ Experiment  <dbl> 1, 0, 0, 1, 0, 1, 1, 0
## $ Pageviews   <dbl> 8669, 8896, 9871, 8448, 9345, 9088, 9308, 8324
## $ Clicks      <dbl> 669, 708, 836, 695, 734, 780, 728, 665
## $ DOW         <fct> Sat, Sun, Tue, Sat, Thu, Fri, Thu, Thu
```

4-Course R-Track for Business Bundle

3.8 Implement Machine Learning Algorithms

We'll implement the new `parsnip` R package. For those unfamiliar, here are some benefits:

- Interfaces with all of the major ML packages: `glmnet`, `xgboost`, `sparklyr`, and more!
- Works well with the `tidyverse` (i.e. `tibbles`)
- Simple API makes Machine Learning easy

Our strategy will be to implement 3 modeling approaches:

1. **Linear Regression** - Linear, Explainable (Baseline)
2. **Decision Tree**
   - Pros: Non-Linear, Explainable.
   - Cons: Lower Performance
3. **XGBoost**
   - Pros: Non-Linear, High Performance
   - Cons: Less Explainable

3.8.1 Linear Regression (Baseline)

We'll create the linear regression model using the `linear_reg()` function setting the mode to "regression". We use the `set_engine()` function to set the linear regression engine to `lm()` from the `stats` library. Finally, we `fit()` the model to the training data specifying "Enrollments" as our target. We drop the "row_id" field from the data since this is a unique ID that will not help the model.

```
model_01_lm <- linear_reg("regression") %>%
   set_engine("lm") %>%
   fit(Enrollments ~ ., data = train_tbl %>% select(-row_id))
```

Next, we can make predictions on the test set using `predict()`. We bind the predictions with the actual values ("Enrollments" from the test set). Then we calculate the error metrics using `metrics()` from the `yardstick` package. We can see that the model is off by about +/-19 Enrollments on average.

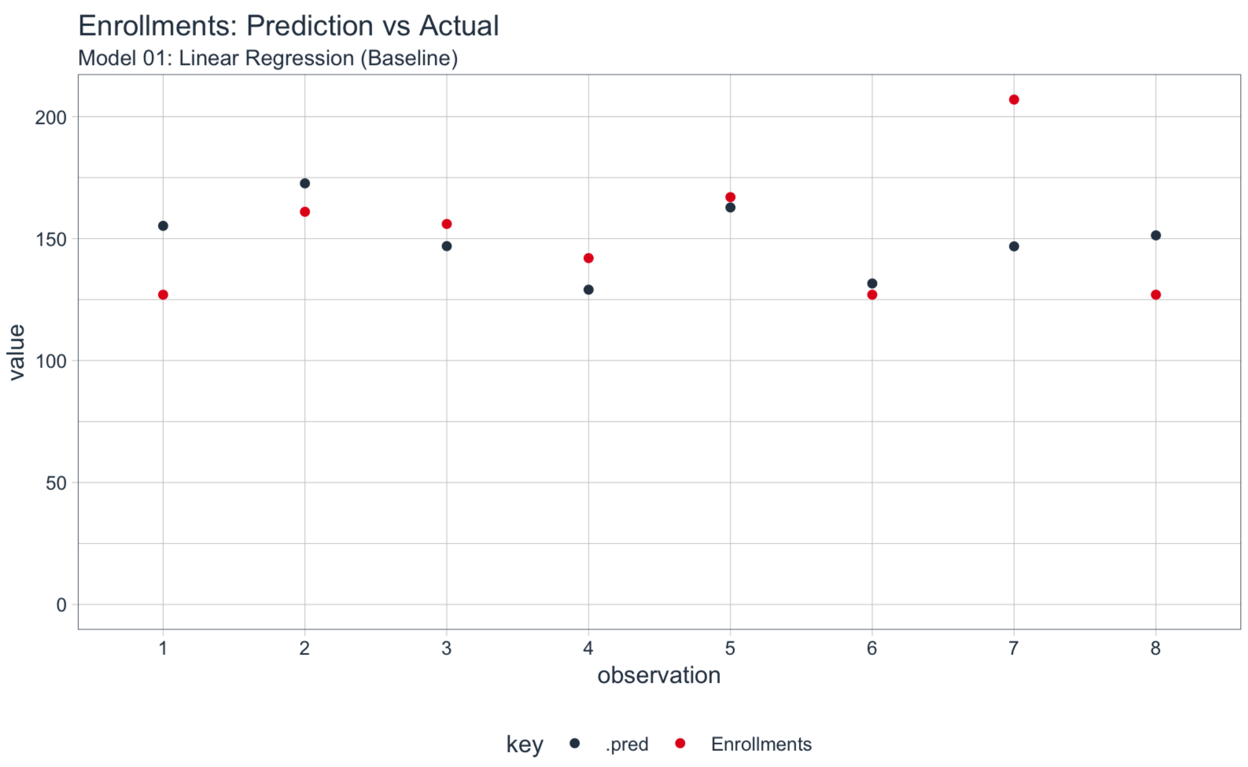
```
# knitr::kable() used for pretty tables
model_01_lm %>%
   predict(new_data = test_tbl) %>%
   bind_cols(test_tbl %>% select(Enrollments)) %>%
   metrics(truth = Enrollments, estimate = .pred) %>%
   knitr::kable()
```

.metric	.estimator	.estimate
rmse	standard	26.0634680
rsq	standard	0.0636897
mae	standard	19.4039158

We can investigate the predictions by visualizing them using `ggplot2`. After formatting and plotting the data, we can see that the model had an issue with Observation 7, which is likely the reason for the low R-squared value (test set).

```
model_01_lm %>%
   # Format Data
   predict(test_tbl) %>%
   bind_cols(test_tbl %>% select(Enrollments)) %>%
   mutate(observation = row_number() %>% as.character()) %>%
   gather(key = "key", value = "value", -observation, factor_key = TRUE) %>%

   # Visualize
   ggplot(aes(x = observation, y = value, color = key)) +
   geom_point() +
   expand_limits(y = 0) +
   theme_tq() +
   scale_color_tq() +
   labs(title = "Enrollments: Prediction vs Actual",
       subtitle = "Model 01: Linear Regression (Baseline)")
```



Ok, so the most important question is what's driving the model. We can use the `tidy()` function from the `broom` package to help. This gets us the model estimates. We can arrange by "p.value" to get an idea of how important the model terms are. Clicks, Pageviews, and Experiment are judged strong predictors with a p-value less than 0.05. However, we want to try out other modeling techniques to judge this. We note that the coefficient of Experiment is -17.6, and because the term is binary (0 or 1) this can be interpreted as decreasing Enrollments by -17.6 per day when the Experiment is run.

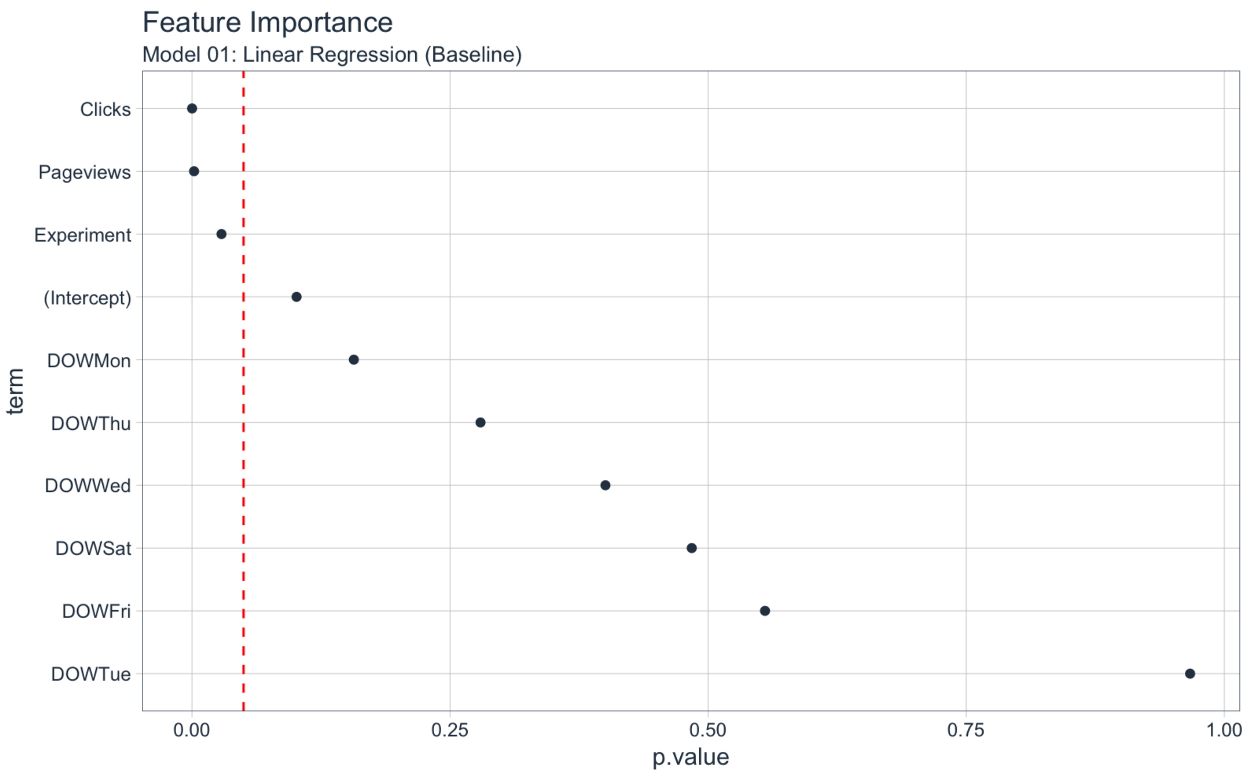
```
linear_regression_model_terms_tbl <- model_01_lm$fit %>%
   tidy() %>%
   arrange(p.value) %>%
   mutate(term = as_factor(term) %>% fct_rev())

# knitr::kable() used for pretty tables
linear_regression_model_terms_tbl %>% knitr::kable()
```

term	estimate	std.error	statistic	p.value
Clicks	-0.5788150	0.1323455	-4.3735162	0.0001533
Pageviews	0.0503213	0.0148314	3.3928928	0.0020803
Experiment	-17.5541754	7.6074013	-2.3075127	0.0286325
(Intercept)	134.7873196	79.5652606	1.6940474	0.1013534
DOWMon	25.4287578	17.4786746	1.4548447	0.1568322
DOWThu	-17.4230930	15.8000716	-1.1027224	0.2795358
DOWWed	12.4235287	14.5532451	0.8536604	0.4005376
DOWSat	-11.0150158	15.5347097	-0.7090584	0.4841509
DOWFri	8.5195293	14.2637448	0.5972856	0.5551166
DOWTue	-0.7023031	16.7661840	-0.0418881	0.9668852

We can visualize the importance separating “p.values” of 0.05 with a red dotted line.

```
linear_regression_model_terms_tbl %>%
   ggplot(aes(x = p.value, y = term)) +
   geom_point(color = "#2C3E50") +
   geom_vline(xintercept = 0.05, linetype = 2, color = "red") +
   theme_tq() +
   labs(title = "Feature Importance",
       subtitle = "Model 01: Linear Regression (Baseline)")
```



Key Points:

- Our model is on average off by +/-19 enrollments (means absolute error). The test set R-squared is quite low at 0.06.
- We investigated the predictions to see if there is anything that jumps out at us. The model had an issue with observation 7, which is likely throwing off the R-squared value.
- We investigated feature importance. Clicks, Pageviews, and Experiment are the most important features. Experiment is 3rd, with a p.value 0.026. Typically this is considered significant.
- We can also see the term coefficient for Experiment is -17.6 indicating as decreasing Enrollments by -17.6 per day when the Experiment is run.

Before we move onto the next model, we can setup some helper functions to reduce repetitive code.

3.8.2 Helper Functions

We'll make some helper functions to reduce repetitive code and increase readability. *Side-Note* - We teach how to create functions and perform iteration in Week 5 of our Business Analysis with R course.

First, we'll make a simplified metric reporting function, `calc_metrics()`.

```
calc_metrics <- function(model, new_data) {
    model %>%
        predict(new_data = new_data) %>%
        bind_cols(new_data %>% select(Enrollments)) %>%
        metrics(truth = Enrollments,
                estimate = .pred)
}
```

Next we can make a simplified visualization function, `plot_predictions()`.

```
plot_predictions <- function(model, new_data) {

   g <- predict(model, new_data) %>%
      bind_cols(new_data %>% select(Enrollments)) %>%
      mutate(observation = row_number() %>% as.character()) %>%
      gather(key = "key", value = "value", -observation, factor_key = TRUE) %>%

      # Visualize
      ggplot(aes(x = observation, y = value, color = key)) +
      geom_point() +
      expand_limits(y = 0) +
      theme_tq() +
      scale_color_tq()

   return(g)
}
```

3.8.3 Decision Trees

Decision Trees are excellent models that can pick up on non-linearities and often make very informative models that compliment linear models by providing a different way of viewing the problem. We teach Decision Trees in Week 6 of Business Analysis with R course.

We can implement a decision tree with `decision_tree()`. We'll set the engine to "rpart", a popular decision tree package. There are a few key tunable parameters:

- `cost_complexity` : A cutoff for model splitting based on increase in explainability
- `tree_depth` : The max tree depth
- `min_n` : The minimum number of observations in terminal (leaf) nodes

The parameters selected for the model were determined using 5-fold cross validation to prevent over-fitting. This is discussed in Important Considerations. We teach 5-Fold Cross Validation in our Advanced Modeling Course - Data Science for Business with R.

```
model_02_decision_tree <- decision_tree(
      mode = "regression",
      cost_complexity = 0.001,
      tree_depth = 5,
      min_n = 4) %>%
   set_engine("rpart") %>%
   fit(Enrollments ~ ., data = train_tbl %>% select(-row_id))
```

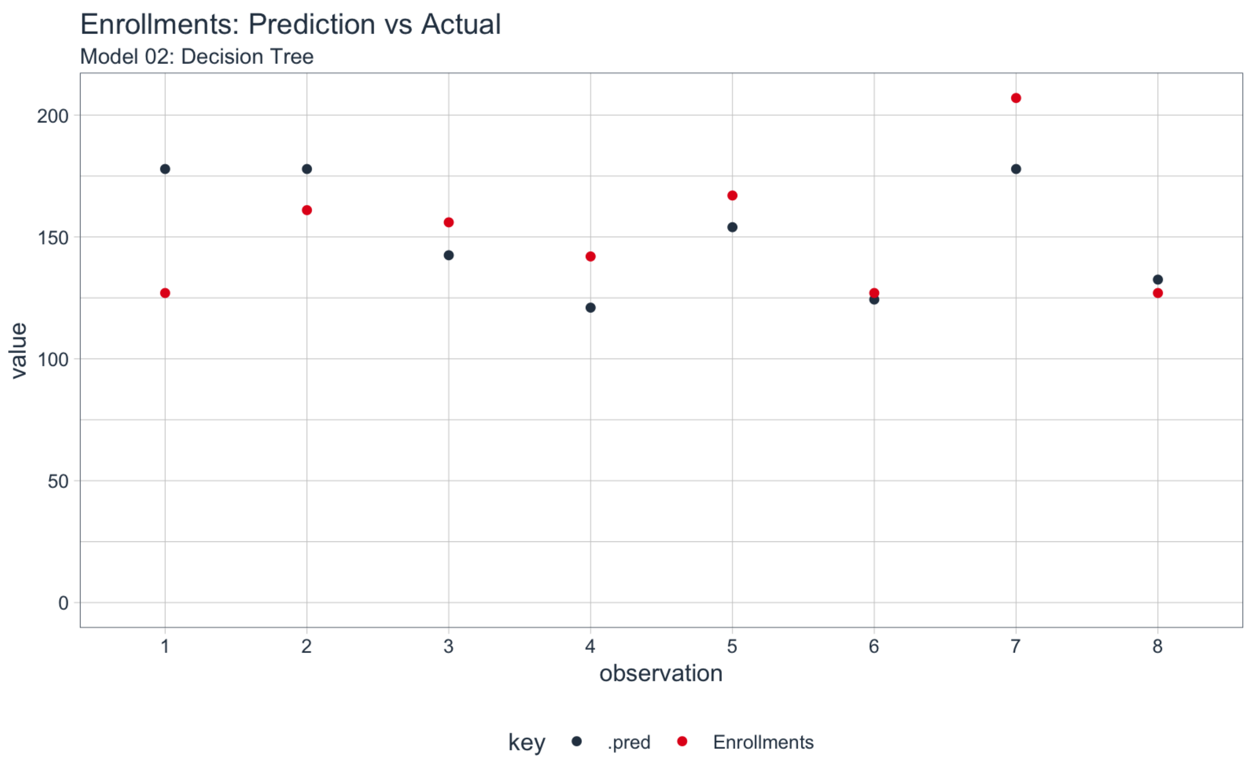
Next, we can calculate the metrics on this model using our helper function, `calc_metrics()`. The MAE of the predictions is approximately the same as the linear model at +/-19 Enrollments per day.

```
# knitr::kable() used for pretty tables
model_02_decision_tree %>%
   calc_metrics(test_tbl) %>%
   knitr::kable()
```

.metric	.estimator	.estimate
rmse	standard	23.85086
rsq	standard	0.27693
mae	standard	19.06771

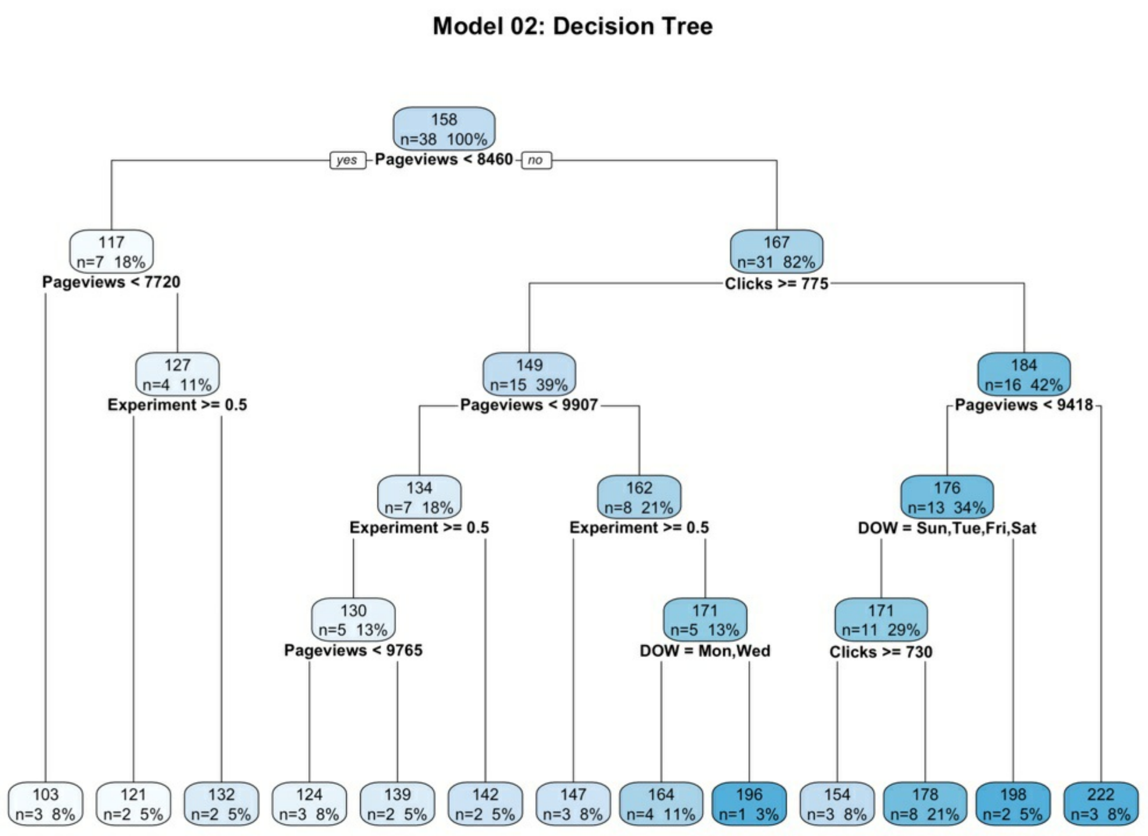
We can visualize how its performing on the observations using our helper function, `plot_predictions()` . The model is having issues with Observations 1 and 7.

```
model_02_decision_tree %>%
   plot_predictions(test_tbl) +
   labs(title = "Enrollments: Prediction vs Actual",
       subtitle = "Model 02: Decision Tree")
```



And finally, we can use `rpart.plot()` to visualize the decision tree rules. Note that we need to extract the underlying "rpart" model from the `parsnip` model object using the `model_02_decision_tree$fit` .

```
model_02_decision_tree$fit %>%
   rpart.plot(
      roundint = FALSE,
      cex = 0.8,
      fallen.leaves = TRUE,
      extra = 101,
      main = "Model 02: Decision Tree")
```



Interpreting the decision tree is straightforward: Each decision is a rule, and Yes is to the left, No is to the right. The top features are the most important to the model ("Pageviews" and "Clicks"). The decision tree shows that "Experiment" is involved in the decision rules. The rules indicate a when Experiment >= 0.5, there is a drop in enrollments.

Key Points:

- Our new model has roughly the same accuracy to +/-19 enrollments (MAE) as the linear regression model.

- Experiment shows up towards the bottom of the tree. The rules indicate a when Experiment >= 0.5, there is a drop in enrollments.

3.8.4 XGBoost

The final model we'll implement is an `xgboost` model. Several key tuning parameters include:

- `mtry` : The number of predictors that will be randomly sampled at each split when creating the tree models.
- `trees` : The number of trees contained in the ensemble.
- `min_n` : The minimum number of data points in a node that are required for the node to be split further.
- `tree_depth` : The maximum depth of the tree (i.e. number of splits).

- `learn_rate` : The rate at which the boosting algorithm adapts from iteration-to-iteration.
- `loss_reduction` : The reduction in the loss function required to split further.
- `sample_size` : The amount of data exposed to the fitting routine.

Understanding these parameters is critical to building good models. We discuss each of these parameters in depth while you apply the ***XGBoost*** model in our Business Analysis with R course.

The parameters selected for the model were determined using 5-fold cross validation to prevent over-fitting. This is discussed in Important Considerations. We teach 5-Fold Cross Validation in our Advanced Modeling Course - Data Science for Business with R.

```
set.seed(123)
model_03_xgboost <- boost_tree(
      mode = "regression",
      mtry = 100,
      trees = 1000,
      min_n = 8,
      tree_depth = 6,
      learn_rate = 0.2,
      loss_reduction = 0.01,
      sample_size = 1) %>%
   set_engine("xgboost") %>%
   fit(Enrollments ~ ., data = train_tbl %>% select(-row_id))
```

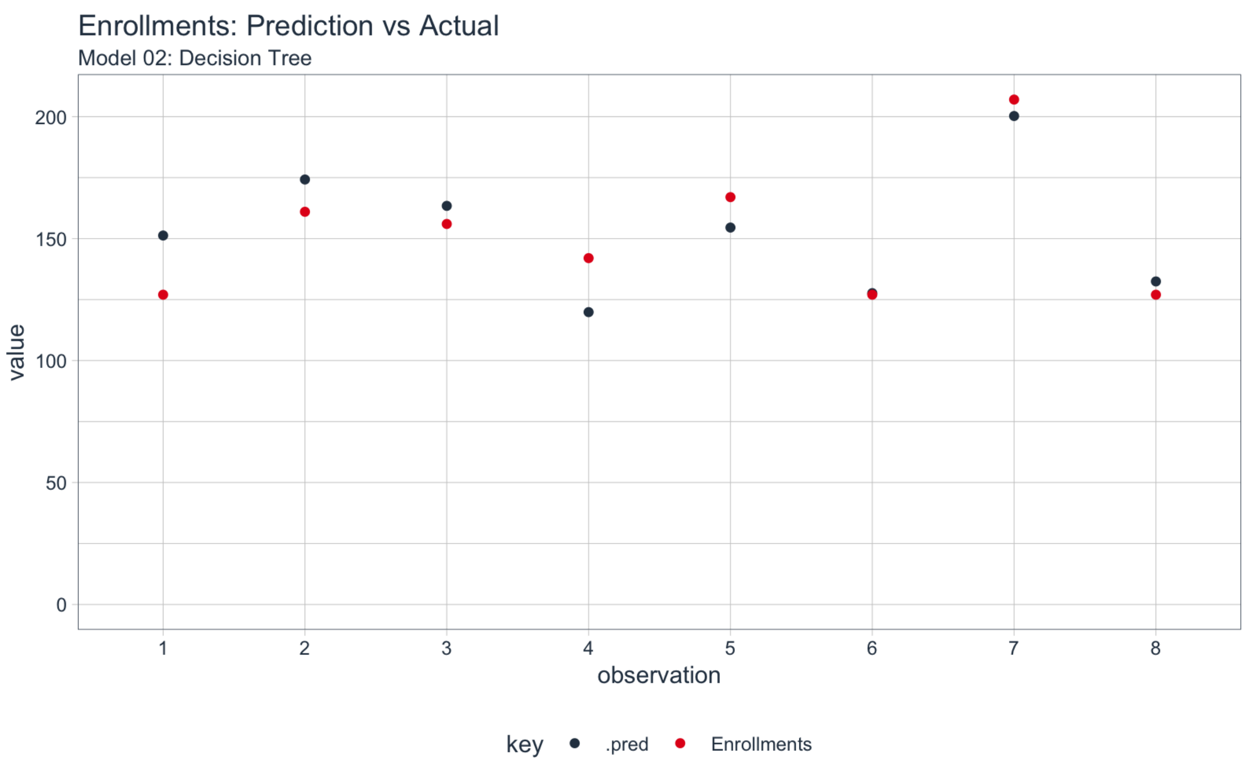
We can get the test set performance using our custom `calc_metrics()` function. We can see that the MAE is 11.5 indicating the model is off by on average 11.5 enrollments per day on the test set.

```
# knitr::kable() used for pretty tables
model_03_xgboost %>%
   calc_metrics(test_tbl) %>%
   knitr::kable()
```

.metric	.estimator	.estimate
rmse	standard	13.8760506
rsq	standard	0.7256724
mae	standard	11.5450172

We can visualize how its performing on the observations using our helper function, `plot_predictions()` . We can see that it’s performing better on Observation 7.

```
model_03_xgboost %>% plot_predictions(test_tbl) +
   labs(title = "Enrollments: Prediction vs Actual",
       subtitle = "Model 02: Decision Tree")
```



We want to understand which features are important to the XGBoost model. We can get the global feature importance from the model by extracting the underlying model from the parsnip object using `model_03_xgboost$fit` and piping this into the function `xgb.importance()` .

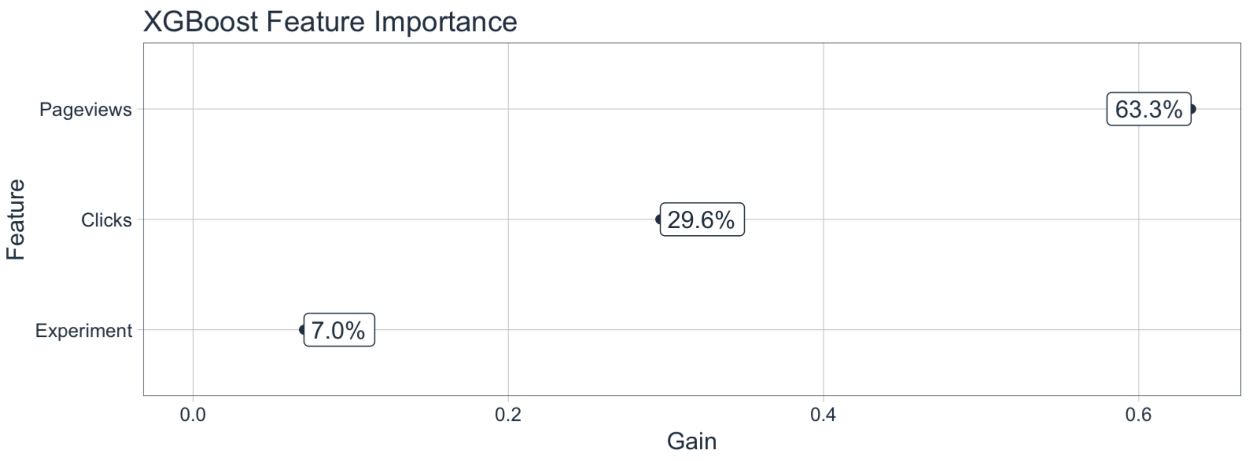
```
xgboost_feature_importance_tbl <- model_03_xgboost$fit %>%
  xgb.importance(model = .) %>%
  as_tibble() %>%
  mutate(Feature = as_factor(Feature) %>% fct_rev())

xgboost_feature_importance_tbl %>% knitr::kable()
```

Feature	Gain	Cover	Frequency
Pageviews	0.6331988	0.6879569	0.6465324
Clicks	0.2964312	0.2061510	0.2076063
Experiment	0.0703701	0.1058921	0.1458613

Next, we can plot the feature importance. We can see that the model is largely driven by Pageviews and Clicks.

```
xgboost_feature_importance_tbl %>%
   ggplot(aes(x = Gain, y = Feature)) +
   geom_point(color = "#2C3E50") +
   geom_label(aes(label = scales::percent(Gain)),
            hjust = "inward", color = "#2C3E50") +
   expand_limits(x = 0) +
   theme_tq() +
   labs(title = "XGBoost Feature Importance")
```



The information gain is 93% from Pageviews and Clicks combined. Experiment has about a 7% contribution to information gain, indicating it's still predictive (just not nearly as much as Pageviews). This tells a story that if Enrollments are critical, Udacity should focus on getting Pageviews.

> This tells a story that if Enrollments are critical, Udacity should focus on getting Pageviews.

Key Points:

- The XGBoost model error has dropped to +/-11 Enrollments.
- The XGBoost shows that Experiment provides an information gain of 7%
- The XGBoost model tells a story that Udacity should be focusing on Page Views and secondarily Clicks to maintain or increase Enrollments. The features drive the system.

3.10 Business Conclusions - Key Benefits to Machine Learning

There are several key benefits to performing A/B Testing using Machine Learning. These include:

- **Understanding the Complex System** - We discovered that the system is driven by Pageviews and Clicks. Statistical Inference would not have identified these drivers. Machine Learning did.

- **Providing a direction and magnitude of the experiment** - We saw that Experiment = 1 drops enrollments by -17.6 Enrollments Per Day in the Linear Regression. We saw similar drops in the Decision Tree rules. Statistical inference would not have identified magnitude and direction. Only whether or not the Experiment had an effect.

What Should Udacity Do?

If Udacity wants to maximimize enrollments, it should focus on increasing Page Views from qualified candidates. Page Views is the most important feature in 2 of 3 models.

If Udacity wants alert people of the time commitment, the additional popup form is expected to decrease the number of enrollments. The negative impact can be seen in the decision tree (when Experiment <= 0.5, Enrollments go down) and in the linear regression model term (-17.6 Enrollments when Experiment = 1). Is this OK? It depends on what Udacity's goals are.

But this is where the business and marketing teams can provide their input developing strategies to maximize their goals - More users, more revenue, and/or more course completions.

3.11 Important Considerations: Cross Validation and Improving Modeling Performance

Two important further considerations when implementing an A/B Test using Machine Learning are:

1. **How to Improve Modeling Performance**
2. **The need for Cross-Validation for Tuning Model Parameters**

3.11.1 How to Improve Modeling Performance

A different test setup would enable significantly better understanding and modeling performance. Why?

1. **The data was AGGREGATED** - To truly understand customer behavior, we should run the analysis on unaggregated data to determine probability of an individual customer enrolling.
2. **There are NO features related to the Customer in the data set** - The customer journey and their characteristics are incredibly important to understanding complex purchasing behavior. Including GOOD features is the best way to improving model performance, and thus insights into customer behavior.

3.11.2 Need for Cross-Validation for Tuning Models

In practice, we need to perform cross-validation to prevent the models from being tuned to the test data set. This is beyond the scope of this tutorial, but is taught in our ***Advanced Machine Learning Course*** - Data Science For Business with R DS4B 201-R.

The parameters for the Decision Tree and XGBoost Models were selected using 5-Fold Cross Validation. The results are as follows.

Model	MAE (Average 5-Fold CV)
Linear Regression	16.2
XGBoost	16.2
Decision Tree	19.2

It's interesting to note that the baseline Linear Regression model had as good of performance (average cross-validation MAE) as XGBoost. This is likely because we are dealing with a simple data set with only a few features.

As we build a better test setup that includes the model performance-boosting recommendations in 3.11.1, I expect that the XGBoost model will quickly take over as the system complexity increases.

4.0 Parting Thoughts and Learning Recommendation

This tutorial scratches the surface of how machine learning can benefit A/B Testing and other multi-million dollar business problems including:

- Customer Churn
- Employee Attrition
- Business Forecasting

> The key is understanding how to construct business problems in the format needed to apply Machine Learning.

We teach these skills at Business Science University. The 2 courses that will ***accelerate*** your data science knowledge are:

- **Business Analysis with R Course (101)** - Designed for beginners and intermediate students. Week 6 covers **Modeling and Machine Learning Algorithms**, which has 44 lessons and 5-hours of video that teaches how to perform machine learning for business using the `parsnip` package (used in this tutorial).

- **Data Science For Business with R Course (201)** - Designed for advanced students that want to learn business consulting combined with advanced data science. **Automatic Machine Learning, Cross Validation, and Hyper Parameter Tuning** are covered in our advanced machine learning course, Week 5, which covers machine learning with `H2O`.

The 2 Courses are integrated, and ***accelerate*** you along your data science journey. We condense years of learning into weeks, which is why the program is so effective.

> Years of learning are condensed into weeks.

For those that need to learn both beginner and advanced machine learning, we have a ***Special 101 + 201 Bundle*** that provides both at an attractive value.

Learn 101 and 201 Combined - Accelerate Your Career Today